CNT4731 Multimedia Networking Principles, Fall 2024
Programming Assignment #3

Due: Please submit your source codes and report (**in PDF format)** via CANVAS by 11:59pm (firm deadline) on Wednesday December 04th

Specifications

The PA#1 was focused on the encoding part of the image, and the PA#2 on download-only client-server communication. This assignment is focused towards the M-JPEG (Motion-JPEG) streaming protocol. For this assignment, you have to write a client-server protocol/program, where the client needs to have a JPEG decoder and is able to successfully display a decompressed image**.**

The server must have a sequence of JPEG compressed files. You can have any number of files, but the minimum must be 10. You have to build a socket/TCP connection between the client and server where the client will fetch the first compressed file and decompress it. The decompressed file is to be displayed for x-ms (x milliseconds) with the use of ***sleep*** command (or the equivalent ones with your OS/IDE).

Note: The client needs to have two buffers, one for the display and other for receiving the next file. The same procedure has to be repeated for the other files. Buffering schemes will be introduced as a lecture on Wednesday November 13th.

For your reference (fps is frames-per-second):

When x is 1000-ms, the display should be 1-fps M-JPEG (and the video lasts 10 seconds).
When x is 500-ms, the display should be 2-fps M-JPEG (and the video lasts 5 seconds).
...
When x is 100-ms, the display should be 10-fps M-JPEG (and the video lasts 1 second).
When x is 50ms, the display should be 20-fps M-JPEG (and the video lasts 0.5 second).

Hints:

1. You do not need to write the JPEG decoder software from the scratch, obtain a stand-alone JPEG decoder in source codes from the public domain can speed up the progress.

2. Once source codes are complied successfully, add a timing measurement via ***gettimeofday*** (or the similar commands with your machine OS/IDE) to do a baseline performance measurement on how fast your JPEG decoder can accomplish the decompression.

3. As you learned from PA#1, the decoding time has to do with the resolution and image complexity too. You thus can estimate the maximal speed of your M-JPEG video display via the control of x-ms.

4. Choose (at least) 10 JPEG compressed images well. **Ideally they should be related, thus M-JPEG can make sense visually**. QCIF resolution is acceptable and CIF or higher is even better (as long as your computer can handle it in realtime).

5. Then create a server program to provide these (at least) 10 JPEG compressed images for your client program. Revise your JPEG decoder into a client program to receive ONE JPEG compressed image at a time. Once the client receives and decompresses the first image into a buffer, it should be displayed for x milli-seconds.

6. Repeat the steps for Image 2 to (at least) Image 10. Human eyes (and brain) will create the motion visually, instead of a slide show (**when the frame rate reached a certain speed)**.

Report Requirements:

1. Please provide a readme with your source code with all the steps provided for running the program. The readme should include your PA3_Team_ID and the steps to run the program.

2. You must create a demo video clip on Youtube. **Please indicate the demo URL/link on your report for me to review the WHOLE demo as the following steps:**

3. Step 1: Show two laptop computers with on machine serving as the server machine, and the other machine serving as the client machine.
4. Step 2: Show the server machine has the source codes and (at least) 10 JPEG images, and the client machine has source codes and ZERO (0) JPEG images.
5. Step 3: Show the compilations on both machines are successful, and executables are ready to launch.
6. Step 4: Show the server process is on, and ready for client to make a connection.
7. Step 5: Show the client process is running **with the displays of 1-fps, 2-fps, 5-fps, 10-fps, 20-fps and max-fps are shown visually.** Note you need to show the client machine has source codes and ZERO (0) JPEG images between every run,

Grading Criteria

Correct Implementation / Outputs	60% (verified via the Youtube demo)
Readability / Comments / Code structures	30%
Report	10%
TOTAL	100%